

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-161985

(43)Date of publication of application : 19.06.1998

(51)Int.Cl.

G06F 15/16

G06F 15/16

G06F 9/46

G06F 12/08

(21)Application number : 08-317496

(71)Applicant : HITACHI LTD

(22)Date of filing : 28.11.1996

(72)Inventor : YAMAUCHI MASAHIKO

YASHIRO HIROSHI

MURAYAMA HIDEKI

HORIYAMA KAZUO

HAYASHI TAKEHISA

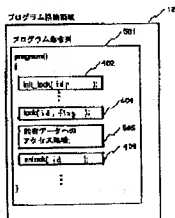
YAMADA KIMITOSHI

(54) PROCESSOR ALLOCATING METHOD, AND MULTIPROCESSOR COMPUTER SYSTEM

(57)Abstract:

PROBLEM TO BE SOLVED: To eliminate useless data transfer between caches and to improve the execution efficiency of a computer system as a whole.

SOLUTION: An instruction string 505 for accessing shared data is sandwiched by a lock acquiring procedure 404 and a lock releasing procedure 406 and any processor is designated by a 2nd argument flag of the lock acquiring procedure 404. A multiprocessor computer system allocates a process for executing the instruction string 505 sandwiched by the lock acquiring procedure 404 and the lock releasing procedure 406 to the processor designated by the 2nd argument flag of the lock acquiring procedure 404. Since the plural processes for accessing the shared data are allocated to the same processor, there is no useless data transfer between caches and the execution efficiency of the entire system can be improved.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

特開平10-161985

(43) 公開日 平成10年(1998) 6月19日

(51) Int. Cl.^{*}
 G 0 6 F 15/16
 9/46
 12/08

識別記号
 3 5 0
 3 8 0
 3 6 0
 3 1 0

F I
 G 0 6 F 15/16
 9/46
 12/08
 3 5 0 A
 3 8 0 Z
 3 6 0 E
 3 1 0 B

審査請求 未請求 請求項の数 3 O L (全 12 頁)

(21) 出願番号 特願平8-317496

(22) 出願日 平成 8 年(1996) 11月28日

(71) 出願人 000005108

株式会社日立製作所
 東京都千代田区神田駿河台四丁目 6 番地

(72) 発明者 山内 雅彦

神奈川県川崎市幸区鹿島田890番地の12
 株式会社日立製作所情報・通信開発本部内

(72) 発明者 巖代 寛

神奈川県川崎市幸区鹿島田890番地の12
 株式会社日立製作所情報・通信開発本部内

(72) 発明者 村山 秀樹

神奈川県川崎市幸区鹿島田890番地の12
 株式会社日立製作所情報・通信開発本部内

(74) 代理人 弁理士 有近 紳志郎

最終頁に続く

(54) 【発明の名称】 プロセッサ割付方法およびマルチプロセッサ計算機システム

(57) 【要約】

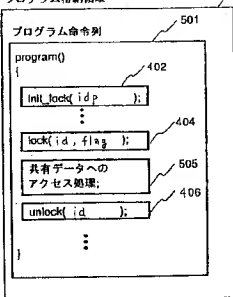
【課題】 キャッシュ間での無駄なデータ転送をなくし、システム全体の実行効率を向上させる。

【解決手段】 共有データにアクセスする命令列 505 をロック獲得手続き 404 とロック解放手続き 406 とで挟むと共にロック獲得手続き 404 の第2 引数 flag でプロセッサを指定する。マルチプロセッサ計算機システムは、ロック獲得手続き 404 とロック解放手続き 406 とで挟まれた命令列 505 を実行するプロセスを、ロック獲得手続き 404 の第2 引数 flag で指定したプロセッサに割り付ける。

【効果】 共有データにアクセスする複数のプロセスを同一のプロセッサに割り付けるため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。

図 5

プログラム格納領域



【特許請求の範囲】

【請求項1】 複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおけるプロセッサ割付方法であつて、

複数のプロセスが共有データにアクセスするとき、それらプロセスを同一のプロセッサに割り付けることを特徴とするプロセッサ割付方法。

【請求項2】 複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおけるプロセッサ割付方法であつて、

ある命令列の前に、プロセッサを指定する相互排除開始命令を配置し、前記命令列の後に、プロセッサの指定を解除する相互排除終了命令を配置し、前記相互排除開始命令でプロセッサが指定されている場合、その指定されたプロセッサにより前記相互排除開始命令と前記相互排除終了命令で挟まれた前記命令列を実行し、前記相互排除開始命令でプロセッサが指定されていない場合、任意のプロセッサにより前記相互排除開始命令と前記相互排除終了命令で挟まれた前記命令列を実行することを特徴とするプロセッサ割付方法。

【請求項3】 複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおいて、

相互排除開始命令によりプロセッサが指定されている場合は、その指定されたプロセッサにより、前記相互排除開始命令から後の命令を実行させ、相互排除終了命令を実行すると、任意のプロセッサにより、前記相互排除終了命令から後の命令を実行させるプロセッサ割り制御手段を具備することを特徴とするマルチプロセッサ計算機システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、プロセッサ割付方法およびマルチプロセッサ計算機システムに関し、更に詳しくは、キャッシュ間のデータ転送回数を減らし、システム全体の実行効率を向上させることができるプロセッサ割付方法およびマルチプロセッサ計算機システムに関する。

【0002】

【従来の技術】 複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおいて、異なるプロセッサに割り付けられた複数のプロセスが共有データにアクセスするとき、共有データの一貫性を保つため、前記異なるプロセスのキャッシュ間で共有データを移動させる制御が行われている。かかる制御の詳細は、UNI Xカーネル内部解析、キャッシュとマルチプロセッサの管理、C. S. Chien著、前川守監訳、岩本信一訳、ソフトバンク株式

会社、293～327頁）に記載されている。

【0003】

【発明が解決しようとする課題】 異なるプロセッサのキャッシュ間で共有データを移動させる制御を行う場合、例えば、異なるプロセッサに割り付けられた各プロセスが交互に共有データに書き込み動作を行なうと、キャッシュ間のデータ転送回数が増え、計算機システム全体の実行効率が下がらな問題点がある。そこで、本発明の目的は、キャッシュ間のデータ転送回数を減らし、システム全体の実行効率を向上させることができるプロセッサ割付方法およびマルチプロセッサ計算機システムを提供することにある。

【0004】

【課題を解決するための手段】 第1の観点では、本発明は、複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおけるプロセッサ割付方法であつて、複数のプロセスが共有データにアクセスするとき、それらプロセスを同一のプロセッサに割り付けることを特徴とするプロセッサ割付方法を提供する。上記プロセッサ割付方法では、共有データにアクセスする複数のプロセスを同一のプロセッサに割り付けるため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。また、複数の共有データがある場合には、各共有データごとに別々のプロセッサを対応させれば、負荷を分散することができる。

【0005】 第2の観点では、本発明は、複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおけるプロセッサ割付方法であつて、ある命令列の前に、プロセッサを指定する相互排除開始命令を配置し、前記命令列の後に、プロセッサの指定を解除する相互排除終了命令を配置し、前記相互排除開始命令でプロセッサが指定されている場合、その指定されたプロセッサにより前記相互排除開始命令と前記相互排除終了命令で挟まれた前記命令列を実行し、前記相互排除開始命令でプロセッサが指定されていない場合、任意のプロセッサにより前記相互排除開始命令と前記相互排除終了命令で挟まれた前記命令列を実行することを特徴とするプロセッサ割付方法を提供する。上記プロセッサ割付方法では、共有データにアクセスする命令列を相互排除開始命令と相互排除終了命令で挟むと共に相互排除開始命令でプロセッサを指定することにより、共有データにアクセスするプロセスを同一のプロセッサに割り付けることが出来る。このため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。また、共有データにアクセスする命令列が複数ある場合には、各命令列ごとに別々のプロセッサを指定すれば、負荷を分散することができる。なお、上記構成において、プロセッサの指定は、プロセッサ番号を明示して

行ってもよいし、マルチプロセッサ計算機システムに一つのプロセッサを選択させることにより行ってもよい。後者の場合、マルチプロセッサ計算機システムは、例えば、相互排除開始命令を実行したプロセッサの番号に基づいてプロセッサを選択したり、共有データが現時点で存在しているキャッシュに対応するプロセッサを選択する。

【0006】第3の観点では、本発明は、複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおいて、相互排除開始命令によりプロセッサが指定されている場合は、その指定されたプロセッサにより、前記相互排除開始命令から後の命令を実行させ、相互排除終了命令を実行すると、任意のプロセッサにより、前記相互排除終了命令から後の命令を実行させるプロセッサ割付制御手段を具備することを特徴とするマルチプロセッサ計算機システムを提供する。上記マルチプロセッサ計算機システムでは、プログラムにおいて共有データにアクセスする命令列を相互排除開始命令と相互排除終了命令とで挟むと共に相互排除開始命令でプロセッサを指定することにより、共有データにアクセスするプロセスを同一のプロセッサに割り付けることが出来る。このため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。また、共有データにアクセスする命令列が複数ある場合には、各命令列ごとに別々のプロセッサを指定すれば、負荷を分散することができる。なお、上記構成において、プロセッサの指定は、プロセッサ番号を明示して行ってもよいし、マルチプロセッサ計算機システムに一つのプロセッサを選択させることにより行ってもよい。後者の場合、プロセッサ割付制御手段は、例えば、相互排除開始命令を実行したプロセッサの番号に基づいてプロセッサを選択したり、共有データが現時点で存在しているキャッシュに対応するプロセッサを選択する。

【0007】

【発明の実施の形態】図1は、本発明の一実施形態にかかるマルチプロセッサ計算機システム101のブロック図である。このマルチプロセッサ計算機システム101は、複数のプロセッサ102(1)~102(N)と、各プロセッサごとのキャッシュメモリ103(1)~103(N)と、命令列やデータを格納するための磁気ディスク等の2次記憶装置104と、メインメモリ106と、システム・バス105とを具備して構成されている。

【0008】前記メインメモリ106には、プロセッサ102が実行するプログラム格納領域120(1)~(N)と、プロセスを管理するためのプロセス管理表117と、プロセッサ102(1)~102(N)で実行中のプロセスを管理するための実行中プロセス管理表116と、プロセッサ102(1)~102(N)には未だ割り付けられていないが実行可能な状態にあるプロセスをリスト構

造で管理する実行可能プロセスリスト118と、プロセッサ102(1)~102(N)に割り付けるプロセスを切り替える手続を格納するプロセス切り替え手続を格納領域119と、複数のプロセスで共有するデータの同一性を保つために他のプロセスが当該共有データを読み書きすることを禁止する相互排除の際に使用する情報を格納するためのロック用データ格納領域110(1)~(M)と、それらロック用データ格納領域110(1)~(M)を初期化する手続を格納するロック初期化手続を格納領域107と、共有データにアクセスする命令列の前で相互排除を指示する手続を格納するロック獲得手続を格納領域108と、共有データにアクセスする命令列の後で相互排除の解除を指示する手続を格納するロック解放手続を格納領域109とがある。

【0009】前記ロック用データ格納領域110(1)~(M)には、対応するデータが相互排除中か否かを示すロック用変数格納領域111と、対応するデータにアクセス中のプロセスがあるか否かを示すロック獲得フラグ格納領域112と、対応するデータにアクセスするプロセスが割り付けられたプロセッサの番号を格納する割付プロセッサ番号格納領域113と、対応するデータにアクセスできる時を待っているプロセスをリストで管理している待機プロセスリスト114とがある。

【0010】図2は、前記プロセス管理表117の構成図である。プロセス管理表117の各行は、それぞれ一つのプロセスに対応している。各行は、行番号を格納するインデックス201と、当該行に対応するプロセスの番号を格納するプロセス番号202と、当該行に対応するプロセスの状態を格納する実行状態203と、当該行に対応するプロセスの実行を中断した時にプログラムカウンタ等のレジスタ情報を退避した記憶領域を格納するレジスタ退避領域204と、実行可能プロセスリスト118のようなリスト構造を作るために使用するリンク先プロセスの行番号(またはリスト構造の末尾を示す識別子END)を格納する次プロセスインデックス205と、当該行に対応するプロセスを割り付けるプロセッサが決まっている場合にその番号を格納する割付プロセッサ番号206とから構成される。

【0011】図3は、前記実行中プロセス管理表116の構成図である。実行中プロセス管理表116の各行は、それぞれプロセッサ102(1)~102(N)に対応している。各行は、当該行に対応するプロセッサ102(1)~102(N)の番号(1)~(N)を格納するプロセッサ番号301と、当該行に対応するプロセッサ102が実行しているプロセスの番号を格納するプロセス番号302とから構成される。

【0012】図4は、相互排除を実行するためにオペレーティング・システムが呼び出す手続であるロック初期化402と、ロック獲得404と、ロック解放406の説明図である。

5

【0013】ロック初期化(手続き名 `init_lock`) 402 は、ロック用データ識別子 `d` を返すための変数格納領域のアドレス `d.p` を引数とする。そして、ロック用データ格納領域110を初期化し、その初期化したロック用データ格納領域110に与えたロック用データ識別子 `d` を前記アドレス `d.p` が指す変数格納領域に格納する機能を有する。

【0014】ロック獲得(手続き名 `lock`) 404 は、前記ロック初期化402で得たロック用データ識別子 `d` を第1引数とし、機能の種類を指定する識別子 `flag` を第2引数とする。そして、ロック用データ識別子 `d` に対応するデータに以後アクセスするプロセスを、第2引数の識別子 `flag` の値がプロセス番号(1~N)なら当該プロセス番号(1~N)に割り付け、識別子 `flag` の値が-1ならシステム側で選択した一つのプロセスに割り付け、さらに、識別子 `flag` の値が-2なら従来技術で提供されているセマフォと同様な相互排除とプロセススケジューリング(プロセスに割り付けるプロセスを選択する処理)を提供するように制御する機能を有する。

なお、セマフォは、P命令とV命令を使って相互排除を実現する。P命令とV命令の詳細は「オペレーティング・システムの機能と構成、高橋他、岩波書店、1983、147~150頁」や「UNI Xカーネルの設計、Maurice 著、坂本他訳、共立出版、1990、334~341頁」に記載されている。また、プロセススケジューリングの詳細は「UNI Xカーネルの設計、Maurice 著、坂本他訳、共立出版、1990、211~219頁」に記載されている。

【0015】ロック解放(手続き名 `unlock`) 406 は、前記ロック初期化402で得たロック用データ識別子 `d` を引数とする。そして、ロック用データ識別子 `d` に対応するデータに以後アクセスするプロセスを任意のプロセスに割り付けることを可能にする機能を有する。

【0016】図5は、ロック初期化402、ロック獲得404、ロック解放406の各手続きを含むプログラム命令列の例示図である。共有データにアクセスする複数のプロセス間で相互排除を行うべき命令列505の前にロック獲得404を置き、後にロック解放406を置く。そして、ロック獲得404より前にロック初期化402を置く。なお、ロック獲得404とロック解放406に挟まれた命令列505は危険領域という。

【0017】図6は、ロック初期化402の処理を示すPAD図である。ステップ601では、ロック用データ格納領域110を確保し、これを構造体 `lock_data` とする。ロック用データ格納領域110のロック用変数格納領域111、ロック獲得フラグ格納領域112、割付プロセス番号格納領域113および特権プロセスリスト114は、それぞれ構造体 `lock_data` のメンバ `lock`、`anybody`、`processor`、`list` となる。ステップ602では、構造体 `lock_data` のメンバ `lock`、`anybody`、`processor`、

6

`list` に初期値を設定する。メンバ `lock` の初期値 `FALSE` は、対応するデータが相互排除されていないことを意味する。メンバ `anybody` の初期値 `NO_PROCESS` は、対応するデータにアクセス中のプロセスがないことを意味する。メンバ `processor` の初期値-1は、対応するデータにアクセスするプロセスがプロセスに割り付けられていないことを意味する。メンバ `list` の初期値は、空(待機プロセスがない)を意味する識別子とする。ステップ603では、ロック用データ構造体 `lock_data` にロック用データ識別子 `d` を与える。ステップ604では、引数のアドレス `d.p` が指す変数格納領域にロック用データ識別子 `d` を格納する。ステップ605では、ロック初期化402の呼び出し元に戻る。

【0018】図7は、ロック獲得404の処理を示すPAD図である。ステップ701では、第1引数のロック用データ識別子 `d` に対応するロック用データ構造体 `lock_data` を得る。ステップ702では、第2引数のフラグ `flag` の値を判定し、値が1~N(プロセス番号)か又は-1(プロセスを一定のプロセスに割り付ける識別子)ならステップ703へ進み、値が-2ならステップ704へ進む。ステップ703では、新規ロック獲得703の手続きを実行する。この新規ロック獲得703の詳細な内容は図8を参照して後述する。

【0019】ステップ704、705では、一命令でメモリに対する読み書きを不可分に実行できるTS(`Test and Set`)命令を使用して、ロック用データ構造体 `lock_data` のメンバ `lock` の値を `true` にする。なお、ステップ705は、他プロセスが危険領域の命令列を実行している場合には、その危険領域の実行が終了するまで待つ処理である。TS命令の詳細は「オペレーティング・システムの機能と構成、高橋他、岩波書店、1983、147頁」に記載されている。TS命令より、ステップ706からステップ707の処理2またはステップ709の処理2までの処理は、ただ一つのプロセスで実行されることが保証される。

【0020】ステップ706では、ロック用データ構造体 `lock_data` のメンバ `anybody` の値が `NO_PROCESS` か否かを判定し、値が `NO_PROCESS` なら(すなわち、危険領域の命令列5を他プロセスが実行していないなら)ステップ707へ進み、値が `NO_PROCESS` でないなら(すなわち、危険領域の命令列を他プロセスが実行しているなら)ステップ709へ進む。

【0021】ステップ707では、処理1で、ロック用データ構造体 `lock_data` のメンバ `anybody` の値を `EXIST_PROCESS` にする。また、処理2で、ロック用データ構造体 `lock_data` のメンバ `lock` の値を `FALSE` にする。ステップ708では、ロック獲得404の呼び出し元に戻る。

【0022】ステップ709では、処理1で、ロック用データ構造体 `lock_data` のメンバ `list` に自プロセスを登

50

録し、他プロセスの実行終了を待つ。処理2で、ロック用データ構造体lock_dataのメンバlockの値をFALSEにする。ステップ710では、処理1で、プロセス管理表117の自プロセスの実行状態203を“中断”にする。処理2で、自プロセスを実行していたプロセスのレジスタの内容を退避し、その退避した領域のアドレスをプロセス管理表117のレジスタ退避領域204に格納する。処理3で、実行中プロセス管理表116から自プロセスのプロセス番号を削除する。ステップ711では、自プロセスを実行していたプロセスに他の“実行可能”状態にあるプロセスを割り付けるために、プロセス切り替え711の手続きを実行する。このプロセス切り替え711の詳細な内容は図1を参照して後述する。ステップ712では、ロック獲得404の呼び出し元に戻る。

【0023】図8は、新規ロック獲得703の処理を示すPAD図である。ステップ801、802では、一命令でメモリに対する読み書きを不可行に実行できるTS命令を使用して、ロック用データ構造体lock_dataのメンバlockの値をtrueにする。なお、ステップ802は、他プロセスが危険領域の命令列を実行している場合には、その危険領域の実行が終了するまで待つ処理である。TS命令により、ステップ801からステップ808の処理3またはステップ812の処理3までの処理は、ただ一つのプロセッサで実行されることが保証される。

【0024】ステップ803では、第1引数で指定されたロック用データ構造体lock_dataのメンバprocessorの値が-1か否かを判定し、-1（プロセスがプロセスに割り付けられていない）ならステップ804に進み、-1でないならステップ807へ進む。ステップ804では、第2引数のフラグflagの値がプロセス番号(1～N)か否かを判定し、プロセス番号(1～N)ならステップ805へ進み、フラグflagの値がプロセス番号でない(-1)ならステップ806へ進む。ステップ805では、フラグflagの値であるプロセス番号(1～N)をロック用データ構造体lock_dataのメンバprocessorの値に設定する。そして、ステップ807へ進む。

【0025】ステップ806では、自プロセスが割り付けられているプロセス番号を実行中プロセス管理表116から調べて、そのプロセス番号をロック用データ構造体lock_dataのメンバprocessorの値に設定する。なお、共有データが現時点で存在しているキャッシュに対応するプロセスの番号を調べて、そのプロセス番号をロック用データ構造体lock_dataのメンバprocessorの値に設定してもよい。そして、ステップ807へ進む。

【0026】ステップ807では、ロック用データ構造体lock_dataのメンバanybodyの値がNO_PROCESSか否か

を判定し、値がNO_PROCESSなら（すなわち、危険領域の命令列を他プロセスが実行していないなら）ステップ808へ進み、値がNO_PROCESSでないなら（すなわち、危険領域の命令列を他プロセスが実行しているなら）ステップ812へ進む。

【0027】ステップ808では、処理1で、ロック用データ構造体lock_dataのメンバanybodyの値をEXIST__PROCESSにする。また、処理2で、ロック用データ構造体lock_dataのメンバprocessorの値をプロセス管理表117の割付プロセス番号206に設定する。また、処理3で、ロック用データ構造体lock_dataのメンバlockの値をFALSEにする。ステップ809では、ロック用データ構造体lock_dataのメンバprocessorの値と自プロセスが割り付けられているプロセス番号とが不一致か否かを判定し、不一致ならステップ810へ進み、一致ならステップ811へ進む。ステップ810では、処理1で、プロセス管理表117の自プロセスの実行状態203を“実行可能”にする。処理2で、自プロセスを実行していたプロセスのレジスタの内容を退避し、その退避した領域のアドレスをプロセス管理表117のレジスタ退避領域204に格納する。処理3で、実行可能プロセスリスト118に自プロセスのプロセス番号を登録する。処理4で、実行中プロセス管理表116から自プロセスのプロセス番号を削除する。ステップ711では、自プロセスを実行していたプロセスに他の“実行可能”状態にあるプロセスを割り付けるために、プロセス切り替え711の手続きを実行する。このプロセス切り替え711の詳細な内容は図1を参照して後述する。そして、ステップ811へ進む。ステップ811では、新規ロック獲得703の呼び出し元に戻る。

【0028】ステップ812では、処理1で、ロック用データ構造体lock_dataのメンバlistに自プロセスを登録し、他プロセスの実行終了を待つ。処理2で、ロック用データ構造体lock_dataのメンバprocessorの値をプロセス管理表117の割付プロセス番号206に設定する。処理3で、ロック用データ構造体lock_dataのメンバlockの値をFALSEにする。ステップ813では、処理1で、プロセス管理表117の自プロセスの実行状態203を“中断”にする。処理2で、自プロセスを実行していたプロセスのレジスタの内容を退避し、その退避した領域のアドレスをプロセス管理表117のレジスタ退避領域204に格納する。処理3で、実行中プロセス管理表116から自プロセスのプロセス番号を削除する。ステップ711では、自プロセスを実行していたプロセスに他の“実行可能”状態にあるプロセスを割り付けるために、プロセス切り替え711の手続きを実行する。このプロセス切り替え711の詳細な内容は図1を参照して後述する。ステップ814では、新規ロック獲得703の呼び出し元に戻る。

【0029】図9は、ロック解放406の処理を示すP

9

AD図である。ステップ901では、新規ロック解放901の手続きを実行する。この新規ロック解放901の詳細な内容は図10を参照して後述する。ステップ902では、引数のロック用データ識別子dに対応するロック用データ構造体lock_dataを得る。ステップ903、904では、一命令でメモリに対する読み書きを不可分に実行できるTS命令を使用して、ロック用データ構造体lock_dataのメンバlockの値をtrueにする。なお、ステップ904は、危険領域の命令列を実行している場合には、その危険領域の実行が終了するまで待つ処理である。TS命令により、ステップ905からステップ906の処理2またはステップ908の処理2までの処理は、ただ一つのプロセッサで実行されることが保証される。

【0030】ステップ905では、ロック用データ構造体lock_dataのメンバflagが空か否かを判定し、空なら(すなわち、他プロセスが待機していないなら)ステップ906へ進み、空でないなら(すなわち、他プロセスが待機しているなら)ステップ908へ進む。

【0031】ステップ906では、処理1で、ロック用データ構造体lock_dataのメンバflagの値にNO_PROCESSを設定する。処理2で、ロック用データ構造体lock_dataのメンバlockの値をFALSEにする。ステップ907では、ロック解放406の呼び出し元に戻る。

【0032】ステップ908では、処理1で、ロック用データ構造体lock_dataのメンバflagから待機しているプロセスの一つを取り出す。処理2で、ロック用データ構造体lock_dataのメンバlockの値をFALSEにする。ステップ909では、処理1で、プロセス管理表117の前記取り出したプロセスの実行状態203を“中断”から“実行可能”に変更する。処理2で、実行可能プロセスリスト118に前記取り出したプロセスのプロセス番号を登録する。処理3で、ロック解放406の呼び出し元に戻る。

【0033】図10は、新規ロック解放901の処理を示すPAD図である。ステップ901aでは、ロック獲得404のステップ808の処理2またはステップ812の処理2で設定したプロセス管理表117の割付プロセス番号206の値がクリアされると、対応するプロセスを任意のプロセッサに割り付け可能となる。ステップ901bでは、新規ロック解放901の呼び出し元に戻る。

【0034】図11は、プロセス切り替え711の処理を示すPAD図である。ステップ1003では、実行可能プロセスリスト118からプロセスの一つを選択する。このとき、実行可能プロセスリストの先頭のプロセスを選択してもよいし、優先度を導入してその優先度の一番高いプロセスを選択してもよい。なお、ステップ1007から戻ってきてステップ1003を再実行したときは、先に選択したプロセス以外のプロセスを選択する。

10

【0035】ステップ1004では、選択したプロセスのプロセス管理表117の割付プロセス番号206が設定されているか否かを判定し、設定されている場合はステップ1005へ進み、設定されていない場合はステップ1009へ進む。ステップ1005では、設定されている割付プロセス番号206と自プロセス(プロセス切り替え711を実行しているプロセッサ)の番号が一致するか否かを判定し、一致した場合にはステップ1009へ進み、一致しない場合にはステップ1007へ進む。ステップ1007では、前記ステップ1003に戻り、先に選択したプロセス以外のプロセスを選択する。

【0036】ステップ1009では、処理1で、前記選択したプロセスを実行可能プロセスリスト118から削除する。処理2で、プロセス管理表117の前記選択したプロセスの実行状態203を“実行可能”から“実行中”に変更する。処理3で、プロセス管理表117の前記選択したプロセスのレジスタ回避領域204により自プロセスのレジスタに内容を復帰する。処理4で、前記選択したプロセスの番号を実行中プロセス管理表116に登録する。処理5で、プロセス切り替え711の呼び出し元に戻る。

【0037】なお、上記プロセス切り替え711は、タイマー割り込み等が発生した時も呼び出される。その時、タイマー割り込み発生を受け取ったプロセッサは、実行中であつたプロセスのレジスタ内容を復元し、そのプロセスの実行状態を“実行中”から“実行可能”に変更し、そのプロセスを実行可能プロセスリストに登録した後、プロセス切り替え711を呼び出す。

【0038】以上のマルチプロセッサ計算機システム101によれば、共有データにアクセスする命令列505をロック獲得404とロック解放406とで挟むと共にロック獲得404の第2引数でプロセス102(1)〜102(N)を指定することにより、共有データにアクセスするプロセスを指定したプロセッサに割り付けることが出来る。このため、キャッシュ103(1)〜103(N)間での無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。

【0039】

【発明の効果】本発明のプロセッサ割付方法およびマルチプロセッサ計算機システムによれば、共有データにアクセスする複数のプロセスを同一のプロセッサに割り付けるため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。

【図面の簡単な説明】

【図1】本発明の一実施形態にかかるマルチプロセッサ計算機システムのブロック図である。

【図2】図1のマルチプロセッサ計算機システムのプロセス管理表の構成図である。

11

【図3】図1のマルチプロセッサ計算機システムの実行中プロセス管理表の構成図である。

【図4】ロック初期化、ロック獲得、ロック解放の各手続きの説明図である。

【図5】ロック初期化、ロック獲得、ロック解放の各手続きを含むプログラム命令列の例示図である。

【図6】ロック初期化手続きの処理を示すPAD図である。

【図7】ロック獲得手続きの処理を示すPAD図である。

【図8】新規ロック獲得手続きの処理を示すPAD図である。

【図9】ロック解放手続きの処理を示すPAD図である。

【図10】新規ロック解放手続きの処理を示すPAD図である。

【図2】

図2.

プロセス管理表	201	202	203	204	205	117	206
インデックス	プロセス番号	実行状態	レジスタ通達領域	実行中プロセス番号	実行中プロセス番号	実行中プロセス番号	実行中プロセス番号
1	1234	実行中					
2	1256	実行中	PL=000001234	5		3	
...
12	1287	中断	PL=00000000		END		
...

【図4】

図4

操作	手続き名	手続きの処理	
		手続きの引数	手続きの戻り値
ロック獲得	lock	1. ロック用データ識別子 id 2. ロック獲得の際、共有データの割り付けられているプロセスに、プロセスを獲得するかどうかが示すフラグ flag	
ロック解放	unlock	1. ロック用データ識別子 id	
ロック初期化	init_lock	1. ロック用データ識別子を指すための初期値格納領域のアドレス idp	

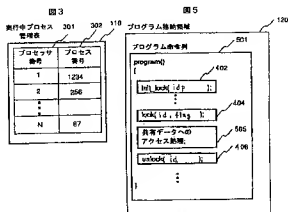
【図11】プロセス切り替え手続きの処理を示すPAD図である。

【符号の説明】

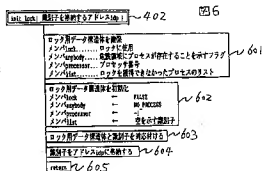
101: マルチプロセッサ計算機システム、102: プロセッサ、103: キャッシュ、104: 2次記憶装置、105: システムバス、106: メインメモリ、107: ロック初期化手続き格納領域、108: ロック獲得手続き格納領域、109: ロック解放手続き格納領域、110: ロック用データ格納領域、111: ロック用変数格納領域、112: ロック獲得フラグ格納領域、113: 割付プロセッサ番号格納領域、114: 待機プロセスリスト、116: 実行中プロセス管理表、117: プロセス管理表、118: 実行可能プロセスリスト、119: プロセス切り替え手続き格納領域、402: ロック初期化手続き(命令)、404: ロック獲得手続き(命令)、406: ロック解放手続き(命令)。

【図3】

【図5】

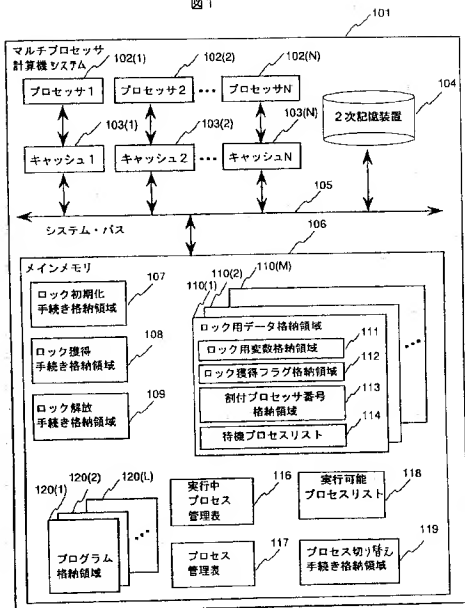


【図6】

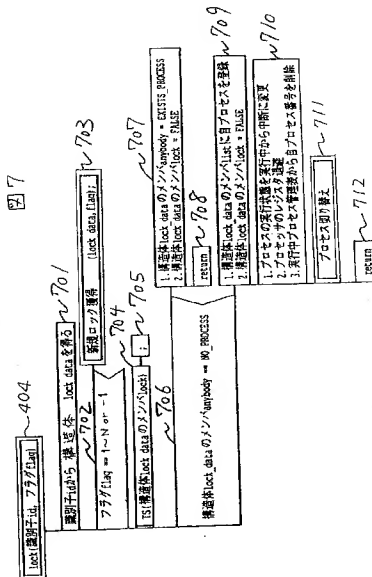


【 図1 】

図1

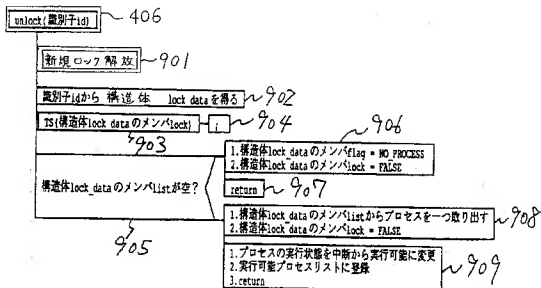


【図7】



【 図9 】

図 9



【 図10 】

図 10

